SCCQL : A Constraint-based Clustering System

Antoine Adam[1], Hendrik Blockeel[1], Sander Govers[2], and Abram Aertsen[2]

[1] KU Leuven, Departement of Computer Science,
Celestijnenlaan 200A, 3001 Leuven, Belgium
[2] KU Leuven, Department of Microbial and Molecular Systems,
Kasteelpark Arenberg 22, 3001 Leuven, Belgium

Abstract. This paper presents the first version of a new inductive database system called SCCQL. The system performs constraint-based clustering on a relational database. Clustering problems are formulated with a query language, an extension of SQL for clustering that includes must-link and cannot-link constraints. The functioning of the system is explained. As an example of use of this system, an application in the context of microbiology has been developed that is presented here.

Keywords: inductive database, inductive query language, clustering

1 Introduction

Data analysis is a non-trivial task: many methods require knowledge of advanced mathematics and statistics to be used correctly, and among those methods that do not, there is still the task of choosing among the many implementations that are available. Data mining environments such as Weka, Orange, RapidMiner, KNIME, etc., facilitate data analysis by allowing the user to construct workflows from predefined building blocks, helping the user choose among alternative techniques, etc. While this provides much support and flexibility, full flexibility is only achieved by allowing scripting or programming in addition to this.

Inductive databases go one step further. Based on the principle that there should be no inherent difference between querying and mining, they offer "data mining query languages", in which data mining tasks can be expressed as queries, and the results are again queriable (the "closure principle"). They set the stage for a more declarative approach to data mining: Just like SQL made it possible to query complex databases without having to program data navigation, inductive query languages should make it possible to formulate complex mining problems without having to choose or compose the optimal mining algorithm. Examples of such systems are SINDBAD/SiQL [6], ATLAS [8], DMX [5].

Constraint-based clustering is an example of a mining task where flexibility is desirable. It is a generalization of standard clustering in which the user can impose constraints on the clustering to be found, such as must-link and cannot-link constraints. Note that also classical parameters of clustering algorithms, such as the number of clusters to be constructed, can be seen as constraints. One could then think of a language that allows the user to naturally formulate "clustering

queries", which may involve a variety of constraints, and of an inductive database system that can execute such queries.

Adam et al. [1] recently proposed such a language. It extends SQL with the CLUSTER statement. The basic structure of this statement is as follows:

```
CLUSTER attributes FROM data [WITH constraints]
```

The currently available constraints are the number of clusters wanted and must-link and cannot-link constraints. These last ones are specified as follows:

```
[SOFT] MUST|CANNOT LINK (cdata) [BY attribute]
```

The complete grammar definition of the language and examples of queries can be found in the aforesaid paper. Note that, while some existing inductive database systems offer clustering queries, none of them offer constraint-based clustering in this manner.

The purpose of this demo is to exhibit a system that allows the user to run this type of query, and to demonstrate the ease with which such a system can be used to solve practical data mining questions. In the remainder of this paper, we briefly describe the architecture of the system and some of its features.

2 The SCCQL system

Figure 1 shows the architecture of the SCCQL ("Structured Constraint-based Clustering Query Language") system, which allows for clustering tuples in a relational database using the mentioned query language. When a cluster query is parsed, the parser does not interpret the `data` and `cdata` parts; these are sent to the actual SQL database, which retrieves the data and sends it to the cluster engine. For soft equivalence constraints, the engine learns a Mahalanobis distance as in [2]. The engine next chooses the algorithm to execute: CopKMeans [7] if there are hard equivalence constraints; the Weka [3] implementation of EM otherwise. Respecting the closure principle, which states that the result of a query should be queriable, the result returned is a table. It is a copy of the `data` table with an extra column holding the cluster assignment of each instance.

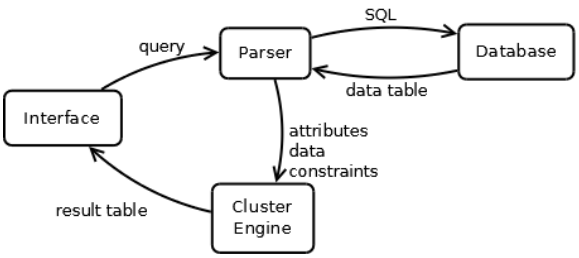

Fig. 1. Architecture scheme.

The system we developed includes an interface that provides two ways of building a query. On one hand, the user can make the query step by step: select the data to cluster, choose the attributes to use for the clustering, specify the number of clusters and add equivalence constraints. The query is then built from the different elements. On the other hand, the user can directly type in the query he wants to execute. The interface also includes some representation of the result table to help visualize it.

3 Application

The SCCQL system is being developed in a project that groups scientists from microbiology and computer science. One of the goals is to build a platform that will help microbiologists to analyse data using data mining techniques. The SCCQL system will be part of this platform.

In the application, clustering has two purposes: finding groups of similar instances, and identifying outliers (instances that are far from any cluster, or in isolated clusters). Such outliers can help microbiologists better understand microbial behavior and growth characteristics. The relational database of the application stores different microbiological parameters of cells, such as the descendancy and physiological states within a population. For instance, one can be interested in clustering cells according to a number of static and/or dynamic parameters. This can be formulated by the following query:

```
CLUSTER LengthMean, WidthMean
FROM (SELECT c.Id, l.Mutant, AVG(s.Length) AS LengthMean,
             AVG(s.Width) AS WidthMean
      FROM stateovertime s, cell c, lineage l
      WHERE l.ExperimentId=5 AND c.LineageId = l.Id
        AND s.CellId = c.Id
      GROUP BY c.id) AS data
WITH SOFT MUST LINK WHERE data.Mutant=0 BY Mutant
```

More examples of queries can be found in [1] and on `http://people.cs.kuleuven.be/~antoine.adam/`.

4 Discussion & Conclusion

A first advantage of the SCCQL system is that it seamlessly integrates the specification of clustering constraints and of the data to be clustered. The latter allows for non-trivial data preprocessing. Not all kinds of preprocessing are easily expressed in SQL, but one could also integrate the system with a language such as SiQL, which does allow more advanced preprocessing. A second advantage is that it is goal-oriented: the user can tell the system what data to cluster, and under what constraints, without stating which clustering method should be used. Although the current implementation uses standard algorithms in its

clustering engine, one could also use general solvers for those cases where no standard algorithm will suffice. The intelligence to chose the system correctly can be built into the system. No other system that we know of combines these two advantages.

For now, SCCQL focuses specifically on constraint-based clustering. Within this setting, its most important limitation is that it can exploit only predefined types of constraints. More types could be introduced (e.g., imposing a minimal cluster size, or requiring balanced clusters), but this will raise the problem of how to solve clustering tasks that combine certain types of constraints. SCCQL is not extensible in the sense of allowing the user to specify any constraint using a general-purpose constraint language. Finally, in the current implementation, the clustering process is not integrated in the database management system: the data is first retrieved from the database, then clustering is performed externally. This is transparent for the user, who just sees the resulting table, but a closer integration may have efficiency advantages.

The SCCQL system is work in progress. We have presented a first version that executes constraint-based clustering queries on a database. An application in the field of microbiology has been shown, but the system is essentially domain independent and can be combined with any SQL database.

Acknowledgements We thank Tias Guns for his useful comments on this paper. This work is funded by the KU Leuven Research Fund (project IDO/10/012).

References

1. Adam, A., Blockeel, H.: A Query Language for Constraint-based Clustering. In Proceedings of the 2013 Belgium-Netherlands Conference on Machine Learning (Benelearn), to appear. (2013)
2. Bar-Hillel, A., Hertz, T. Shental, N., Weinshall, D.: Learning a Mahalanobis Metric from Equivalence Constraints. Journal of Machine Learning Research 6(1), 937-965 (2005).
3. Hall, M., Frank, E. Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter, 11(1), 10-18 (2009)
4. Imieliński, T., Mannila, H.: A Database Perspective on Knowledge Discovery. Communication of the ACM 39.11, 58-64 (1996)
5. Data Mining eXtensions DMX, `http://msdn.microsoft.com/en-us/library/ms132058.aspx`
6. Wicker, J., Richter, L., Kessler, K., Kramer, S.: SINDBAD and SiQL: An Inductive Database and Query Language in the Relational Model. In: W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp.690-694. Springer-Verlag Berlin Heidelberg (2008)
7. Wagstaff, K.L.: Intelligent Clustering with Instance-level Constraints. PhD diss., Cornell University (2002)
8. Wang, H., Zaniolo, C.: Atlas: A Native Extension of SQL for Data Mining. In: Proceedings of the 3rd SIAM International Conference on Data Mining (2003)